AMENDMENTS TO THE CLAIMS

Please amend the claims as indicated in the following listing of all claims:

1. (Currently amended) A method of profiling code for an execution environment in which latency exists between an execution event and detection thereof, the method comprising:

executing the code on a processor;

detecting the execution event; and

backtracking from a detection point in the code coinciding with the detection of the execution event to a preceding operation associated with the execution event, the backtracking identifying the preceding operation at ~~an expected~~ a predetermined displacement from the detection point unless an ambiguity creating location is disposed therebetween, wherein the predetermined displacement is dependent upon the preceding operation which is indicated by the execution event.

2. (Original) The method of claim 1,

wherein the ambiguity creating location is a branch target location.

3. (Currently amended) The method of claim 2,

wherein the ambiguity creating location ~~otherwise associated with at least some branch target locations~~ is bridged using branch history information.

4. (Original) The method of claim 1,

wherein the ambiguity creating location is an entry point location.

5. (Original) The method of claim 1, wherein the ambiguity creating location is one of:

a jump target location;

an indirect branch target location;

a trap handler location; and

an interrupt handler location.

6. (Original) The method of claim 1,

wherein the preceding operation corresponds to a load instruction; and
wherein the execution event is a cache miss.

7. (Original) The method of claim 1,
wherein the preceding operation corresponds to a memory access instruction; and
wherein the execution event is either a hit or a miss at a level in a memory hierarchy.

8. (Currently amended) The method of claim 1,
wherein the execution event is either [[a]] an overflow or an underflow of a hardware counter.

9. (Currently amended) The method of claim 1,
wherein the execution event triggers either [[a]] an overflow or an underflow of a hardware counter that is itself detected.

10. (Original) The method of claim 1,
wherein the latency includes that associated with delivery of a trap.

11. (Original) The method of claim 1,
wherein the latency includes that associated with delivery of a counter overflow event signal.

12. (Original) The method of claim 1,
wherein the latency is associated with pipeline execution skid.

13. (Original) The method of claim 1,
wherein the latency is associated with completion of in-flight operations.

14. (Original) The method of claim 1, embodied in a computer program product.

15. (Original) The method of claim 1, embodied in at least one of:
a profiling tool;

a code optimizer; and

a runtime library.

16. (Original) The method of claim 1, employed in combination with a compiler that pads the code with one or more padding operations to absorb at least some instances of the latency.

17. (Original) The method of claim 16,

wherein the padding operations are not themselves associated with the execution event.

18. (Original) The method of claim 16,

wherein the padding operations are not themselves ambiguity creating locations.

19. (Currently amended) A method of identifying operations associated with execution events of a processor, the method comprising:

from a point in an execution sequence of the operations of the processor, the point coinciding with an execution event, backtracking through the operations toward a particular operation that precedes the coinciding point by ~~an expected~~ a predetermined latency; and

associating the execution event with the particular operation unless the backtracking encounters an unresolved intervening target of a control transfer, wherein the predetermined latency is dependent upon the particular operation which is indicated by the execution event.

20. (Currently amended) The method of claim 19, further comprising:

executing the sequence of operations on [[a]] the processor; and

detecting the execution event.

21. (Currently amended) The method of claim 19,

wherein the operations are instructions executable on [[a]] the processor; and

wherein the particular operation is a particular one of the instructions that triggers the execution event.

22. (Currently amended) The method of claim 19,

wherein the operations ~~are executable on a processor and~~ correspond to instructions of program code.~~; and~~

~~wherein the particular operation corresponds to a particular one of the operations that triggers the execution event.~~

23. (Original) The method of claim 19,

wherein the execution event is an exception triggering execution of the particular operation.

24. (Original) The method of claim 19,

wherein the execution event is a cache miss.

25. (Currently amended) The method of claim 19,

wherein the expected latency includes [[an]] a trap delivery delay.

26. (Original) The method of claim 19,

wherein the execution event triggers a hardware event and the expected latency includes delivery of a signal associated therewith.

27. (Original) The method of claim 26,

wherein the hardware event is either underflow or overflow of a counter associated with the execution event.

28. (Original) The method of claim 19,

wherein the execution event is either underflow or overflow of a counter.

29. (Original) The method of claim 19,

wherein instances of intervening control transfer targets are identified in the execution sequence of operations to facilitate the backtracking.

30. (Original) The method of claim 29,

wherein at least some of the instances of intervening control transfer targets are resolved using branch history information.

31. (Original) The method of claim 19,
wherein control transfer target locations in the execution sequence are identified by a compiler.

32. (Currently amended) The method of claim 19,
wherein the operations are instructions executable on [[a]] the processor; and
wherein the particular operation is a particular one of the instructions that triggers the execution event.

33. (Currently amended) The method of claim 19,
wherein the operations ~~are executable on a processor and~~ correspond to instructions of program code.~~; and~~
~~wherein the particular operation corresponds to a particular one of the operations that triggers the execution event.~~

34. (Original) The method of claim 19, further comprising:
preparing the execution sequence of the operations.

35. (Original) The method of claim 34,
wherein preparation of the execution sequence includes identifying a location of the control transfer target therein.

36. (Original) The method of claim 34,
wherein preparation of the execution sequence includes identifying a location of the particular operation therein.

37. (Original) The method of claim 19,
wherein the particular operations include memory referencing instructions.

38. (Original) The method of claim 37,

wherein the memory referencing instructions include one or more of loads, stores and prefetches.

39. (Currently amended) A method of associating an execution characteristic of code executed on a processor with a particular operation thereof, the method comprising:

identifying at least first-type and second-type operations in the code;

from a point in an execution sequence of the code that coincides with delayed detection of the execution characteristic, backtracking a predetermined displacement toward a candidate triggering operation of the first-type and associating the candidate triggering operation with the execution characteristic unless an unresolved intervening operation of the second-type is encountered, wherein the predetermined displacement is dependent upon the first-type operations which are indicated by the execution characteristic.

40. (Original) The method of claim 39,

wherein the first-type operations include memory access operations.

41. (Original) The method of claim 39,

wherein the second-type operations include operations that coincide with control transfer target locations in the code.

42. (Original) The method of claim 39,

wherein the execution characteristic involves memory access latency.

43. (Original) The method of claim 39,

wherein the execution characteristic includes a cache miss statistic.

44. (Original) The method of claim 39,

wherein the detection delay includes a pipelined execution skid latency.

45. (Original) The method of claim 39,

wherein the second-type operations include operations that coincide with branch target locations in the code; and

wherein at least some of the branch target locations are resolved using branch history information.

46. (Currently amended) A method of preparing code for a processor, the method comprising:

preparing a tangible first executable instance of the code, the preparing identifying at least ambiguity creating locations therein;

executing the first executable instance and responsive to detection of an execution characteristic, backtracking a predetermined displacement through the code to identify an associated operation thereof, wherein extent of the backtracking is limited at least by encountering of an unresolved intervening one of the identified ambiguity creating locations, wherein the predetermined displacement is dependent upon the associated operation which is indicated by the execution characteristic; and

further preparing a tangible second executable instance of the code using the association between the associated operation and the execution characteristic.

47. (Currently amended) The method of claim 46,

wherein the association between the associated operation and the execution characteristic is based on a ~~statistically-significant~~ set of additional detections and responsive backtracking.

48. (Original) The method of claim 46,

wherein the execution characteristic involves memory access latency; and

wherein the preparation of the second executable instance includes insertion of prefetch operations into the code.

49. (Original) The method of claim 46, further comprising:

resolving at least some intervening ones of the identified ambiguity creating locations using branch history information.

50. (Currently amended) A computer program product encoded in one or more computer readable media, the computer program product comprising:

an execution sequence of operations; and

padding operations following at least some particular operations of the execution sequence, the padding operations providing an unambiguous skid region of the execution sequence between the at least some particular operations and subsequent ambiguity creating locations to provide a predetermined latency between the ambiguity creating locations and the at least some particular operations to facilitate determination of which of the at least some particular operations are associated with execution events.

51. (Original) The computer program product of claim 50,

wherein the particular operations include memory access operations.

52. (Original) The computer program product of claim 50,

wherein the padding operations include nops.

53. (Original) The computer program product of claim 50,

wherein the unambiguous skid region does not include an ambiguity creating location.

54. (Currently amended) The computer program product of claim 50,

wherein the one or more computer readable media are selected from the set of a disk, tape, ~~or other~~ magnetic, optical, semiconductor or electronic storage medium and a network, wireline, ~~wireless~~ or ~~other~~ wireless communications medium.

55. (Currently amended) A computer program product encoded in one or more computer readable media, the computer program product comprising:

an execution sequence of operations; and

one or more data sections that identify in the execution sequence at least ambiguity creating locations and target operations for use by one or both of a profiler and a optimizer, wherein the target operations occur at predetermined displacements from execution events unless at least one of the at least ambiguity creating

locations is disposed therebetween, and wherein the predetermined displacements are dependent upon the target operations which are indication by the execution events.

56. (Original) The computer program product of claim 55,
wherein the ambiguity creating locations include branch target locations.

57. (Original) The computer program product of claim 55,
wherein the target operations include memory referencing operations.

58. (Currently amended) The computer program product of claim 55,
wherein the one or more computer readable media are selected from the set of a disk, tape, ~~or other~~ magnetic, optical, semiconductor or electronic storage medium and a network, wireline, ~~wireless~~ or ~~other~~ wireless communications medium.

59. (Currently amended) An apparatus comprising:
means for backtracking a predetermined placement, from a point coinciding with an execution event in an execution sequence of operations on a processor, through the execution sequence toward a particular operation thereof that precedes the coinciding point; and
means for associating the execution event with the particular operation unless the backtracking encounters an intervening ambiguity creating location, wherein the predetermined displacement is dependent upon the particular operation which is indicated by the execution event.

60. (Original) The apparatus of claim 59, further comprising:
means for bridging at least some ambiguity creating locations.

61. (Currently amended) An apparatus comprising:
a code preparation facility suitable for preparation of an execution sequence of operations for a processor; and

means for padding the execution sequence to provide an unambiguous skid region between a particular operation and a subsequent ambiguity creating location within the sequence of operations to provide a predetermined latency between the ambiguity creating location and the particular operation to facilitate determination of which of the sequence of operations is associated with an execution event~~therein~~.